Continuous-Time Recurrent Neural Networks for Generative and Interactive Musical Performance

Oliver Bown[1] and Sebastian Lexer[2]

[1] Centre for Cognition, Computation and Culture,
[2] Department of Music,
Goldsmiths College, University of London,
New Cross, SE14 6NW, UK
o.bown@gold.ac.uk, s.lexer@incalcando.com

Abstract. This paper describes an ongoing exploration into the use of Continuous-Time Recurrent Neural Networks (CTRNNs) as generative and interactive performance tools, and using Genetic Algorithms (GAs) to evolve specific CTRNN behaviours. We propose that even randomly generated CTRNNs can be used in musically interesting ways, and that evolution can be employed to produce networks which exhibit properties that are suitable for use in interactive improvisation by computer musicians. We argue that the development of musical contexts for the CTRNN is best performed by the computer musician user rather than the programmer, and suggest ways in which strategies for the evolution of CTRNN behaviour may be developed further for this context.

1 Introduction

At the junction between computer music and artificial intelligence lies the goal of developing generative or interactive software agents which exhibit musicality. The appearance of musicality is determined either by a listening, watching audience or an interacting performing musician. Attempts in this domain have taken on a wide variety of forms due, on the one hand, to the wide variety of methods in artificial intelligence, and broadened further by the myriad possible interpretations of these techniques in a musical domain, and in addition by the myriad musical styles and substrates in which such an interpretation takes place [16, 10].

The approach presented here is inspired first and foremost by a search for general-purpose behavioural entities that could be adopted by computer musicians in a flexible manner. Our approach is influenced by, and indeed made possible by, the availability and popularity of modular extensible computer music platforms such as Max/MSP [2], PD [3] and SuperCollider [1]. Practising musicians who work with these tools often build up personalised repertoires of software patches and commonly adapt publicly available third-party objects to their own performance needs. This feeds a powerful form of social creative search in which something designed with a given purpose in mind may be re-appropriated indefinitely. Thus rather than thinking in terms of stand-alone intelligent musical systems, we conceive of a generic behavioural tool that can be developed in different directions by practising musicians.

This paper proposes the Continuous-Time Recurrent Neural Network (CTRNN) as one such tool. The remainder of this section gives a background and technical description of the CTRNN and discusses aspects of its behaviour that are relevant to musical improvisation. Section 2 discusses the implementation of the CTRNN in Max/MSP and initial performance uses of the CTRNN, and expands upon our methodology. Sections 3 and 4 discuss methods for evolving CTRNNs and for designing performance contexts for them.

1.1 Background

Non-symbolic artificial intelligence (AI) emphasises low-level behaviours at the heart of all species' strategies for survival, as understood in terms of Darwin's theory of evolution [6]. Early work in cybernetics by Grey-Walter [7] established an experimental context in which wheeled robots, containing sensors and motors connected by simple analogue circuits, could be designed to produce observably *lifelike* behaviour. Since the development of Genetic Algorithms (GAs) and increasingly smaller and faster computer processors, it has become possible to evolve compact algorithms that allow a physical wheeled robot to satisfactorily perform more precisely defined cognitive tasks.

The notion of *minimal cognition* [13, 5] has helped home in on the meaning of the term *lifelike*. In recent years a great effort has been made to understand how extremely simple biologically-inspired algorithms could learn tasks such as object recognition, selective attention and simple memory, using CTRNNs *embodied* in simulated agents and *situated* in simple physical environments.

1.2 Technical Description of the CTRNN

CTRNNs are a kind of artificial neural network: an interconnected network of simulated neurons modelled on a computer. In the case of CTRNNs neurons are typically of a type known as the *leaky integrator*. This is a greatly simplified model of a real neuron, with a continually updating internal state determined by a differential equation,

$$\tau_i(dy_i/dt) = -y_i + \sum W_{ij}\sigma(g_j(y_j - b_j)) + I_i \tag{1}$$

where τ_i is the time constant, g_i is the gain and b_i is the bias for neuron i, I_i is any external input for neuron i and W_{ij} is the weight of the connection between neuron i and neuron j. σ is a non-linear transfer function which in our case is *tanh*.

CTRNNs allow recurrency, meaning that network connections can exist in any direction, including potential connections from any node to itself. The combination of recurrency and internal state makes for a system which can produce complex internal patterns of activity and which has a memory-like response to

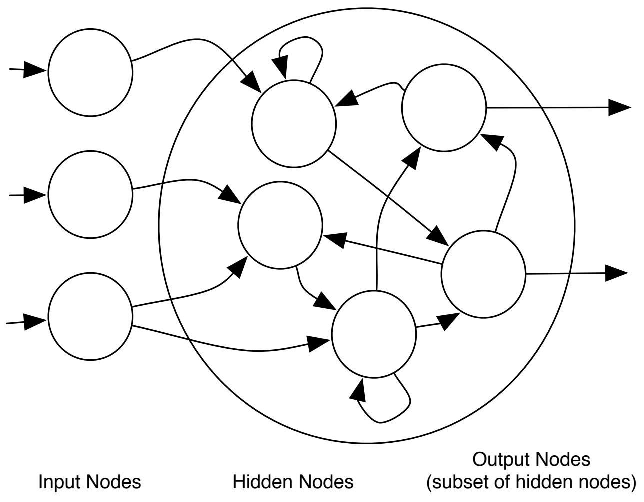

Fig. 1. CTRNN architecture

its environment [13]. Each node has three parameters – bias, gain and time constant – associated with its behaviour, and each connection between nodes has one parameter – a weight. Due to the complex relation between network parameters and network behaviour, along with the non-specificity of solutions to tasks that they are typically used for, a common method for arriving at a CTRNN which performs a certain task is to use a Genetic Algorithm (GA).

CTRNNs are known to be theoretically capable of replicating any dynamical system, and it has been shown that very small CTRNNs are capable of arbitrarily complex dynamics [4].

1.3 Categorisation of Behaviour from CTRNN

Our interest in the CTRNN as a musical unit stems from the potentially unbounded range of temporal dynamics of which it is capable. It is a sub-symbolic system, meaning that it is unlikely to have immediate application in the domain of discretised musical events traditional to computer music and fundamentally implicit in human musical behaviour. Although an appropriate use of CTRNNs would therefore be in the signal domain, we focus on simple rhythmic behaviours at the control rate, and we refer to this domain as gestural. The CTRNN's interaction with the world consists of vectors of real valued inputs and outputs, updating continually and frequently, in our case on the order of 10 milliseconds. A representative example of the CTRNN in a musical context, which illustrates our intended use of the system, places it with a series of features extracted from an audio stream as input, and a set of synthesis parameters as output. In this case the CTRNN is conceived of as a direct interactive participant in a musical performance.

Beer [4] provides an extensive mathematical analysis of the behaviour of small CTRNNs, the presentation of which is beyond the scope of this paper. A key notion in such an analysis is that nodes with suitably strong self-connections can *saturate*, meaning that their own feedback dominates their input and locks them in a certain state. In this way nodes can act as internal switches which influence the behaviour of the rest of the network. Such nodes may flip states. More broadly, due to its internal state and recurrency, the state of a CTRNN at time t is determined not only by the present input, but the history of inputs up until time t, and the starting state of the network.

In the case of a static input (*i.e.*, whose values are not changing), CTRNNs can be described in terms of their internal dynamics in the same way as any closed dynamical system. The network will either find a static resting state, move periodically, or move quasiperiodically or chaotically [9].

In the case of changing inputs, we consider three distinct categories of CTRNN behaviour *as perceived.* These categories are specific to our musical purpose and are not derived from a formal approach: they attempt to capture a musician's perception of CTRNN behaviour rather than CTRNNs themselves.

In the first category, each input state leads to one output pattern, which may be either static or cyclical. As the input state moves from A to B and back again, the CTRNN moves between associated patterns, returning to the same pattern for each input.

The second category is identical to the first except that after changing input state from A to B, a transitory period of indeterminate length precedes the output's arrival at its resting pattern. The length and form of these transitionary sections vary depending on the particular trajectory through input states, but always end up at the same pattern for any given input state.

In the third category, there may be more than one output pattern for each input state, and which one is arrived at will be determined by the trajectory through input states. Thus moving from input state A to input state B leads to a different output pattern than if one were to move via input state C. In other words, the system has multiple attractors for the same resting input, and is dependent on the history of the input leading up to its resting state.

2 Musical Uses

The use of neural networks in studies of music cognition and in composition already has a long and rich history. Commonly cited benefits of a neural network approach to musical problems are *generalisation*, the possibility of extrapolating general features from a corpus of examples, and *graceful degradation*, the robust nature of the network's response to unexpected inputs, as compared to rule-based systems. [15] and [8] contain an exemplary cross-section of such work. Mozer [11], for example, uses trained recurrent neural networks to generate musical melodies with note-by-note prediction. He identifies and addresses problems of securing musical structure over longer time scales than are naturally dealt with by the network, and thus improves the quality of melodies generated in this way. In this and other work in music and AI the final goal is often a machine that makes novel competent music within a given context without the aid of a musician. According to the valuable 'Frankenstein' analogy provided by Todd and Werner in the same volume [16], the present approach differs from this body of work by beginning with the problem of the 'monster's' acceptance in society.

2.1 The CTRNN as Max/MSP Object

The first author has implemented the CTRNN in C as a Max/MSP external object. The object can read and write networks to and from plain text files, and can also randomly generate networks according to a set of user-specified parameters. It can also save and recall network states.

Inputs are sent to the CTRNN as a list of floats to the object's left inlet. Each input list triggers one update of the network, causing a list of output values to be sent from the object's left outlet. Inputs should therefore be sent at equal time intervals. For behaviour suitable for human interaction, a rate of the order of 10 milliseconds is appropriate. This can easily be achieved in a robust manner in Max/MSP.

We describe below how networks with specific behaviour have been generated using a GA in a simple command-line program so that the Max/MSP object can then read them. In other cases, users can randomly generate networks by selecting a number of input nodes, a number of outputs, and a number of internal (hidden) nodes, as well as setting maximum and minimum values for a number of parameters that will then be generated at random. These ranges include values for the time constants, biases and gains for both hidden nodes and input nodes (separate ranges are allowed for each type of node), for weights of connections from input nodes to hidden nodes, and finally for weights of connections from hidden nodes to hidden nodes. There is also a *density* parameter which determines the proportion of connections that are non-zero. Randomly generated networks obviously do not have precisely pre-specified behaviour, but they still obey tendencies. For example it is easy to vary the number of hidden nodes and the density of networks and observe an increased activity as both of these values are increased.

2.2 Development of a Methodology

Initial motivation for using the CTRNN came from a desire to generate metrically free rhythmic patterns, inspired by the work of Karl Sims in evolving locomotive behaviours [12]. Rhythmic patterns were achieved by placing a threshold on one of the CTRNN outputs in order to trigger drum events. This idea was the subject of the first author's MSc dissertation, and was conducted entirely in a non-realtime context.

The second author, working with freely improvised electroacoustic music, has started working with a trial version of the CTRNN Max/MSP object to control a spectral filter through which he plays the piano. This reframes the original motivation behind using the CTRNN according to a distinction in electroacoustic theory between the systems autonomy of the instrument and the control of the performer. According to this the most interesting period in the case of the piano is its decay period, which lies beyond the control of the player: the sound is on its own after the hammer strikes the string. The pianist John Tilbury describes this as the *contingency* of the piano sound [14]. The present form of the CTRNN, even randomly generated, offers an opportunity to implement similar contingent relationships between performer and electroacoustic process, generating micro structures that control electroacoustic processes in turn dependent on the performer's activity. Given the CTRNN as a Max/MSP object the user would be free to develop his or her own approach to the problem of how to map inputs and outputs, and take on the responsibility of making the CTRNN sound good in his or her own musical context through an iterated approach of performance and adjustment. The challenge of refining parameter mappings is already familiar to any musician developing interactive software.

Introducing evolution to the project suggests the possibility of developing a system that has certain behavioural facets that would drive an engaging improvisation in very basic ways: for example responding to an input pattern but not in the same way every time; behaviour potentially changing over time; settling into patterns but producing variations on themes. Put more loosely, such a system should have idiosyncrasies and tendencies that the user could put to effective use to drive an improvisation. However, writing fitness functions that result in these desires being met is not straightforward. In the following section we describe initial attempts to do this.

3 Evolving Musical CTRNNs

Our long term aim is to develop methods for allowing musician users to be able to define their own behavioural targets for CTRNNs. However, since the use of GAs is not straightforward, some thought needs to be given to how this is facilitated, and what kind of behaviours can be defined. This paper deals with a simple initial experiment to evolve CTRNNs that exhibit the third category of behaviour in section 1.3, in the hope that the resulting networks exhibit behaviour that is inherently appealing to musicians.

This only goes half way to the claim of evolving musical behaviour since there are many musical considerations left over. In the previous section, we argued that the CTRNN should be placed in a musical context by the musician user. In the language of behavioural robotics, this means it is up to the musician to embody and situate the CTRNN. Embodiment, in this context, refers to the CTRNNs set of input and output systems in Max/MSP, such as audio analysis tools at the input, and synthesiser parameters at the output. Situatedness, in this context, refers to the musical environment in which the CTRNN will be used, which includes the playing styles and knowledge of the performers who will be playing with the CTRNN. Note that whilst, ideally, behavioural systems are evolved in embodied, situated contexts, the complications that come with achieving this in a musical context are *side-stepped* by trying to evolve general-purpose musical behaviours. Such issues are addressed in the concluding remarks.

CTRNNs are assumed to be fully-connected in the hidden layer, and with a full set of connections from the input layer to the hidden layer. Non-fully connected CTRNNs are thus expressed as having some zero weights. The number of parameters needed to describe a fully-connected CTRNN with n hidden nodes and m input nodes is $n^2 + 3n + nm + 3m$ (since there are n^2 connections between the hidden nodes, and nm connections from the input nodes to the hidden nodes, each with a single weight, as well as bias, gain and time constant parameters for each of the hidden and input nodes). Genotypes for the CTRNNs were expressed as vectors of real numbers in the range $\{0,1\}$, with a mapping from genotype to phenotype that transformed this range to prespecified ranges for each of the parameters. All mappings were linear except for time constants, which were mapped exponentially.

A rank-based GA was used with standard crossover and mutation. At each generation, pairs of individuals from the fittest third of the population were selected at random for breeding, and their offspring were used to replace individuals from the weakest third of the population. Mutation involved adding a random vector of average length 0.01, drawn from a Gaussian distribution, to the individual genotype string. Genotype values that fell outside of the range $\{0,1\}$ were *bounced* back in with an elasticity of 0.1. Multipoint crossover was used, with a probability of 0.1 that the source genotype would be swapped at each point along the genotype.

The fitness function for the CTRNN was as follows: a number x of input patterns were generated using random walks starting from the origin (the same set of input patterns were then used for all of the trials). For each input pattern the CTRNN was reset to zero, and run on the input values for t time steps, then on a fixed input of zero for another t time steps, and then for another $t/10$ time steps, still with the zero input, the output sequence for this final period being stored. The x stored output sequences were then compared with each other in order to establish their similarity. For each pair of outputs, absolute differences between corresponding pairs of values in the output sequences were taken and summed to produce a dissimilarity score. This was repeated with different time offsets, and the lowest possible dissimilarity score was taken (*i.e.*, the score for the case in which the pairs were most similar). The average of the dissimilarity scores for each pair of outputs was taken as the fitness of the CTRNN. This meant that out of phase, but otherwise identical, periodic outputs would receive a score of zero. Two versions of the evolutionary process were attempted. In the first a gradual increase in the number x of input sequences over the course of the GA was used in order to smooth the difficulty of the task. Early on in the GA CTRNNs had tasks involving 3 input sequences, and over time this was increased to 20 input sequences. In the second version, 20 input sequences were used right from the beginning. GAs were run for 2000 generations. t was fixed at 500.

A number of evolutionary runs were made with different values for the number of hidden nodes from 3 to 10. In each case networks were give two inputs and three outputs. This was so that the network could be later interacted with using a two-dimensional controller in Max/MSP (hence the two inputs) and could be visualised in a three-dimensional parameter-space plot (hence the three outputs).

The resulting CTRNNs were tested for the generality of their behaviour by being compared with 200 random CTRNNs of the same number of nodes on an identical task as the fitness function, but with a new set of random input patterns which were different from the ones used in the fitness function. In other words, the CTRNNs were tested to see how far their history-dependent behaviour extended. Figure 2 shows the results of such a test averaged over 10 trials. A few random nodes score higher than the evolved nodes. It was noted that random nodes with a higher density of connections scored higher, exhibiting more erratic behaviour in general. However, despite the fact that the performance of the evolved CTRNNs is quite variable, the graph indicates that the evolved behaviour has some generality.

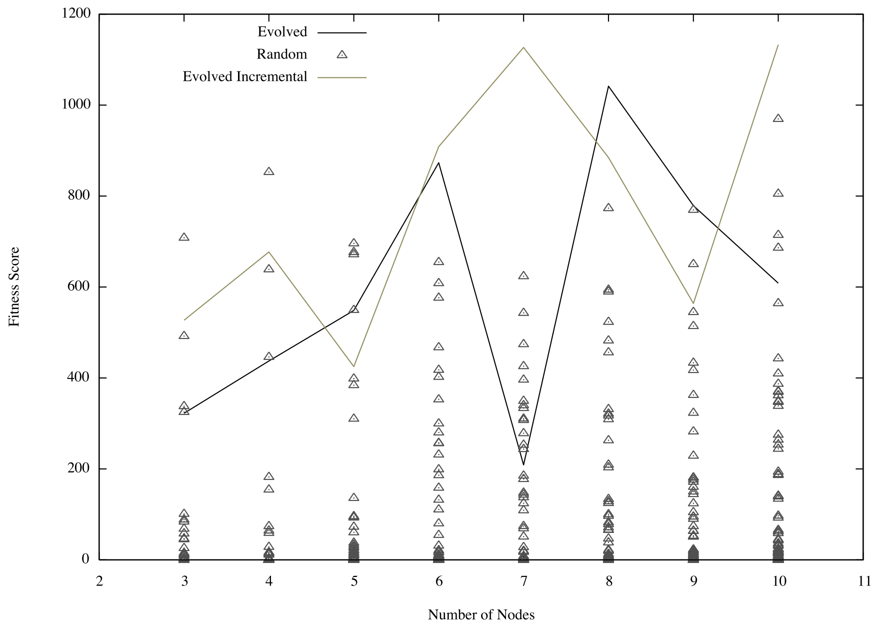

Fig. 2. Generality of fitness of evolved networks (lines) versus random networks (points) for a range of node numbers. The lighter line shows networks evolved using an incremental fitness function

Informal interactive testing of the evolved and random networks showed a remarkable range of behaviours, and the most immediate implication of this is that a more thorough categorisation of CTRNN behaviour from a musical point of view is in order. Evolved networks tended to be doing something more *interesting* and often the nature of their evolved behaviour was immediately apparent: by repeating the same input patterns one could easily observe the network falling into distinct cyclic attractors. The incrementally evolved networks seemed to find less satisfying solutions which often involved oscillating at a rate near to the time step of the CTRNN update rules; a behaviour that it would probably be beneficial to punish. Larger networks were generally more interesting, but the best networks from this trial, as judged by the first author, were the 5 and 10 node evolved networks, suggesting that larger networks are not necessarily more interesting.

Similar informal tests with a wider set of CTRNNs revealed the potential for compositionally useful behaviour in almost all networks, both random and evolved: regardless of interactive potential or sustained varied dynamics, many of the oscillations made by CTRNNs at cyclic attractors were rhythmically and dynamically pleasing when connected up to various synthesis or filter parameters.

We have used the word 'interesting' without hope of qualifying it with *why* things are interesting at this stage. More formal tests should be set up to determine which kind of network behaviours are of interest to a range of performing musicians. Since our focus is on interactive behaviour in an improvisational context our main concern is whether the CTRNNs can sustain interest through interaction. Whilst our small evolved CTRNNs had a relatively predictable behaviour (although they produced more sustained interest than random networks of a similar size), more complex CTRNNs (with large numbers of densely connected nodes) produced complex output patterns that were intriguing to listen to but appeared to have very little to do with what was being played to them, thus failing to be interactive. Truly interesting interactive behaviour comes with the sense that the CTRNN is responding to its input *at the same time as* performing autonomously. This has been hard to pinpoint in the present study but should be made a central concern in future tests.

We conclude that it is relatively easy to define targets for CTRNN behaviour and approach those targets through evolution, but that further investigation is needed into *how* to specify target behaviour that is musically useful. Now that a framework is in place in which CTRNNs can be evolved and tested, both in simulation and through direct interaction in musical contexts, it will be possible to extensively explore a variety of approaches to designing fitness functions and applying network behaviours to musical goals. Plans for future work in these areas are discussed in the following section.

4 Future Work

4.1 Finding Target Behaviours

With the standalone GA application in place, exploring behaviours evolved for different tasks becomes a simple matter of writing new fitness functions. The

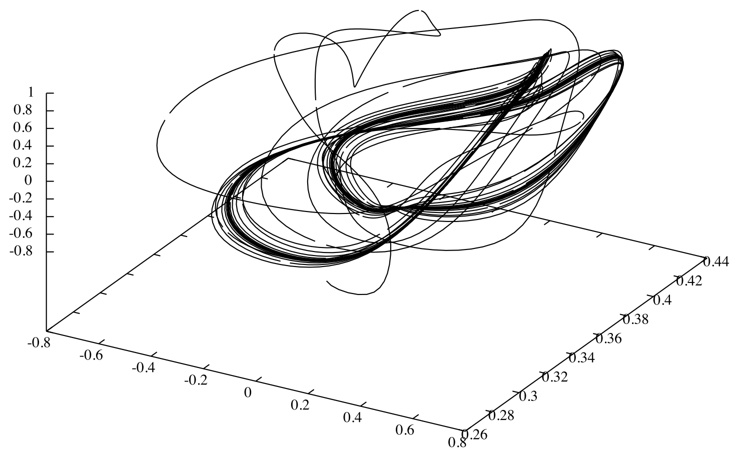

Fig. 3. Example of an output trajectory for an evolved CTRNN

fitness function used in the experiment above aims to produce a behaviour that exhibits some minimal aspect of musicality. A range of similarly general purpose fitness functions could be added to this to produce a repertoire of behavioural units, and to this extent it would be desirable to test a variety of simple fitness functions in an iterative manner with improvising musicians. Alternative functions could focus on levels of predictability, rhythmic features of the CTRNN behaviour, call and answer dynamics, or producing specific patterns under specific circumstances.

A natural development would be to get CTRNNs to imitate human performances, beginning with recorded training data which the CTRNN is expected to imitate. Referring back to our representative example in section 1.3, a performer could take over the part of the CTRNN, controlling the filter, and all input and output data could then be recorded. However, this approach implies difficulties: it would not be sufficient just to feed the CTRNN with the input sequences and rate it according to how close its output comes to the target output sequences. At the very least the sequences would need to be divided into a set of discrete trials; this is necessary in order to convince the network to pay any attention to its inputs. To this end it will be necessary to explore how data sets can be recorded and divided into significant events either manually or automatically.

4.2 Developing Mappings from CTRNN Outputs

When using the CTRNN in Max/MSP, the user is able to observe three dimensions of output states from a CTRNN in a window generated in Jitter, the video editing extension to Max/MSP. By observing output states during practice we propose developing this interface so that a musician could draw colour-coded regions into the 3-D space that he or she wishes to correlate to specific parameter settings of an instrument. A feedforward neural network could then be trained to implement the desired mapping. Through an iterative process of practice and adjustment a more carefully crafted combination of behaviour and desired sound could be developed, bringing together a CTRNN behaviour with a specific repertoire of output states. Successfully implementing this addition may reinforce the notion that it is sufficient to provide the musician with a set of CTRNNs with general-purpose behaviours. The musician is then able to chose from a set of behaviours, and iteratively design a mapping from behaviour to audible musical output. Similar processes could be applicable to the input of the network.

4.3 Evoking a Coevolutionary Context

Better still would be to create a context in which the behaviour of the CTRNN can be modified in real time, so that the CTRNN, its embodiment (the Max/MSP input-output context), and its situatedness (the musical context that it will be used in) could collectively converge, rather than the latter two converging around the former. Thus experimentation with real-time interactive CTRNNs begs the notion of a coevolution between user and unit that would lead to a powerful interactive system at the moment of performance, but would also imply a gradual adaptive development of all aspects of the system during preparation. CTRNNs can also be modified in various ways to introduce *ontogenic adaptation* (as opposed to evolution) into the preparation process. This paper stops short of suggestions for how these developments could be achieved, but research into this problem could take various immediate directions likely to throw up fruitful results.

5 Summary

We have introduced the CTRNN as a performative and/or compositional tool for musicians using modular extensible computer music platforms such as Max/MSP. We have described how networks can be randomly generated or evolved to produce particular behavioural properties, and demonstrate very simple examples in which evolved CTRNNs exhibit behaviours that are of interest to improvising musicians.

We have discussed future work in this area, including gathering training data to be used for the evolution of more specific CTRNN behaviours and developing mappings from CTRNNs to performance parameters using a trained feedforward network. We suggested that the CTRNN should be adapted by musicians according to their own performance contexts and their own interpretation of its behaviour, and that it should inform their own actions during performance as well as during the development of their performance contexts.

The notion of a coevolution or adaptive codevelopment between CTRNN behaviour and user is provoked by the present work. We suggest that this problem could be made into a fruitful topic of research.

Acknowledgments

Oliver Bown's research is supported by a bursary from the Department of Computing, Goldsmiths College. We would like to thank Geraint Wiggins for extensive feedback and Mike Riley and the Goldsmiths Department of Visual Arts' Digital Media Lab for the use of their computers as Apple XGrid agents.

References

1. http://www.audiosynth.com.
2. http://www.cycling74.com.
3. http://www.puredata.info.
4. R. D. Beer. On the dynamics of small continuous recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.
5. R.D. Beer. The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209–243, 2003.
6. C. Darwin. *On the origin of species by means of natural selection, or The preservation of favoured races in the struggle for life.* D. Appleton and company, 1860.
7. W. Grey Walter. An imitation of life. *Scientific American*, 182(4):42–54, 1950.
8. N. Griffith and P. M. Todd, editors. *Musical Networks.* MIT Press, 1999.
9. D. Kaplan and L. Glass. *Understanding Nonlinear Dynamics.* Springer-Verlag, 1995.
10. E. Miranda. *Composing Music with Computers.* Focal Press, 2001.
11. M. C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.
12. K. Sims. Evolving 3d morphology and behaviour by competition. In *Artificial Life IV Proceedings.* MIT Press, 1994.
13. A.C. Slocum, D.C. Downey, and R.D. Beer. Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. In J. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 430–439. MIT Press, 2000.
14. J. Tilbury. Feldman and the piano: the art of touch and celebration of contingency. In *Second Biennial International Conference On Twentieth-Century Music, Goldsmiths College, University of London*, 2001.
15. P. M. Todd and D. Gareth Loy. *Music and Connectionism.* MIT Press, 1991.
16. P. M. Todd and G. Werner. Frankensteinian approaches to evolutionary music composition. In Niall Griffith and Peter M. Todd, editors, *Musical Networks: Parallel Distributed Perception and Performance*, pages 313–339. MIT Press/Bradford Books, Cambridge, MA, 1999.

This article was processed using the LaTeX macro package with LLNCS style